

Topic: Machine vision and artificial intelligence

**A quantitative analysis on the effectiveness of CNN based method
compared to Optical Flow based method for frame interpolation**

Research question:

To what extent are CNN based methods better than optical-flow based methods
for interpolating videos with lower frame rate?

Computer Science Extended Essay

4002 Words

CS EE World
<https://cseeworld.wixsite.com/home>
27/34 (A)
May 2024

Submitter info:
Hey Osten here! I am studying CS at uni.
Feel free to contact me at [ostenantonio1 \[at\] gmail \[dot\] com](mailto:ostenantonio1@gmail.com)

Table of contents

1. Introduction.....	1
1.1 Hypothesis.....	2
2. Background Information.....	3
2.1. Optical Flow.....	3
2.1.1. Lucas-Kanade Method.....	4
2.1.2. Horn-Schunck method.....	5
2.2. Deep Learning.....	7
2.3. Convolutional Neural Networks (CNN).....	8
3. Methodology.....	9
3.1. Testing Data Used.....	9
3.2. Dependent Variable.....	10
3.2.1. Mean Squared Error (MSE).....	10
3.2.2. Structural Similarity Index (SSIM).....	10
3.2.3. Peak Signal to Noise Ratio (PSNR).....	11
3.2.4. Learned Perceptual Image Patch Similarity (LPIPS).....	11
4. Results.....	13
4.1. Linear Motion.....	13
4.1.1. Tabular data representation.....	13
4.1.2. Visual representation.....	16
4.1.3. Data analysis.....	17
4.2. Rotary Motion.....	20
4.2.1. Tabular data representation.....	20
4.2.2. Visual representation.....	23
4.2.3. Data analysis.....	24
4.3. Oscillatory Motion.....	26
4.3.1. Tabular data representation.....	26
4.3.2. Visual representation.....	29
4.3.3. Data analysis.....	30
5. Overall results.....	33
6. Conclusion.....	37
6.1. Future research opportunities.....	38
7. Bibliography.....	39
7.1. Programs used.....	41
Code for getting dependent variable.....	42
Code for getting the difference mask.....	44

List of Tables and Figures

Tables

Table 1. Linear fast speed (1080p).....	13
Table 2. Linear fast speed (240p).....	13
Table 3. Linear medium speed (1080p).....	14
Table 4. Linear medium speed (240p).....	14
Table 5. Linear slow speed (1080p).....	14
Table 6. Linear slow speed (240p).....	15
Table 7. Average linear results.....	15
Table 8. Rotary fast speed (1080p).....	20
Table 9. Rotary fast speed (240p).....	20
Table 10. Rotary medium speed (1080p).....	21
Table 11. Rotary medium speed (240p).....	21
Table 12. Rotary slow speed (1080p).....	21
Table 13. Rotary slow speed (240p).....	22
Table 14. Average rotary results.....	22
Table 15. Oscillatory fast speed (1080p).....	26
Table 16. Oscillatory fast speed (240p).....	26
Table 17. Oscillatory medium speed (1080p).....	27
Table 18. Oscillatory medium speed (240p).....	27
Table 19. Oscillatory slow speed (1080p).....	27
Table 20. Oscillatory slow speed (240p).....	28
Table 21. Average oscillatory result.....	28
Table 22. Average program results (resolution).....	36
Table 23. Average program results (speed).....	36
Table 24. Difference in program results (speed).....	37

Figures

Figure 1. Optical flow of a Barber's pole.....	3
Figure 2. Neighboring pixels of $N \times N$ pixels (Lucas-Kanade method).....	4
Figure 3. Three partial derivatives of the image brightness (Horn-Schunck method).....	5
Figure 4. Example for deep learning's interconnected nodes.....	7
Figure 5. Example for CNN area topology.....	8
Figure 6. Simplified view of the LPIPS network.....	11
Figure 7. Linear fast speed (1080p) PSNR graph.....	16
Figure 8. Linear medium speed (1080p) PSNR graph.....	16
Figure 9. Linear slow speed (1080p) PSNR graph.....	16
Figure 10. Linear fast speed (240p) PSNR graph.....	16
Figure 11. Linear medium speed (240p) PSNR graph.....	16
Figure 12. Linear slow speed (240p) PSNR graph.....	16
Figure 13. Comparison, frame 22, Linear slow speed (1080p).....	17
Figure 14. Comparison, frame 22, Linear fast speed (240p).....	17
Figure 15. Zoomed in comparison with difference mask, frame 22, Linear fast speed (240p)....	18
Figure 16. Comparison (CNN), frame 39, Linear fast speed (1080p).....	19
Figure 17. Rotary fast speed (1080p) PSNR graph.....	23
Figure 18. Rotary medium speed (1080p) PSNR graph.....	23
Figure 19. Rotary slow speed (1080p) PSNR graph.....	23
Figure 20. Rotary fast speed (240p) PSNR graph.....	23
Figure 21. Rotary medium speed (240p) PSNR graph.....	23
Figure 22. Rotary slow speed (240p) PSNR graph.....	23
Figure 23. Comparison with difference mask, frame 13, rotary fast speed (1080p).....	24
Figure 24. Zoomed in comparison, frame 32, rotary slow speed (240p).....	25
Figure 25. Oscillatory fast speed (1080p) PSNR graph.....	29
Figure 26. Oscillatory medium speed (1080p) PSNR graph.....	29
Figure 27. Oscillatory slow speed (1080p) PSNR graph.....	29
Figure 28. Oscillatory fast speed (240p) PSNR graph.....	29
Figure 29. Oscillatory medium speed (240p) PSNR graph.....	29
Figure 30. Oscillatory slow speed (240p) PSNR graph.....	29
Figure 31. Comparison, frame 22, oscillatory medium speed (1080p).....	30
Figure 32. Zoomed in comparison, frame 34, oscillatory fast speed (1080p).....	31
Figure 33. Comparison, frame 30, oscillatory fast speed (1080p).....	31
Figure 34. Comparison with difference mask, frame 13, rotary fast speed (1080p).....	32
Figure 35. Comparison between programs, transparency effect.....	33
Figure 36. Comparison between programs, disappearance effect.....	34
Figure 37. Zoomed in comparison, frame 6 to 12, Rotary slow speed (1080p).....	35

1. Introduction

Video frame interpolation is a technique that has revolutionized the world of video processing. Video frame interpolation is the process of synthesizing several artificial frames in the middle of two adjacent frames of the original video [1]. It is being used in many video processing purposes, such as the creation of slow-motion videos [2], video stabilization [3], frame rate conversion, and video restoration [4]. Recently, it is used in other fields like the medical field, where the technique is used to get a coronary angiography video with a high frame rate without the risk of harming both the patients and surgeons [1].

To be able to interpolate videos, two input frames are needed, they are then utilized to analyze how frames move from one to another. There are numerous ways of synthesizing artificial frames but the most used method among all of them are CNN based methods, and optical-flow based methods [5].

Both methods have their own advantages and disadvantages, so this research seeks to investigate the extent of which method is better than the other, specifically which one performs better on several metrics such as SSIM (Structural Similarity), MSE (Mean Squared Error), and PSNR (Peak Signal-to-Noise Ratio).

This research will be useful in the future especially as future technologies and contents can become more advanced, thus requiring a more efficient integration of video frame interpolation. Especially with the development of many methods of video frame interpolation, this research can help improve the performance of them by pointing out the disadvantages of the two most prevalent methods, which can subsequently optimize and improve the development and quality of both future algorithms (by providing the limitations

and drawbacks of current methods) and contents (with improved algorithms, the quality of contents can also be enhanced). Moreover, the findings of this research can also benefit other industries such as the medical and entertainment industries [6].

To investigate this relationship, different programs that use different methods of interpolating (optical-flow, and deep-learning) will be tested, and they will be made to interpolate several video frames from different videos. The generated frames will be then analyzed and compared against each other using the aforementioned metrics (SSIM, MSE, and PSNR).

1.1 Hypothesis

According to the research [4], it suggests that CNN-based methods perform slightly worse than Optical-flow methods. However, with consideration across the datasets used in the research, the average for Optical-flow is only slightly better than CNN-based methods. Hence, the main hypothesis for this research is that Optical-flow will perform better than CNN-based methods in interpolating videos with very low frame rate, but the extent is not that large.

2. Background Information

2.1. Optical Flow

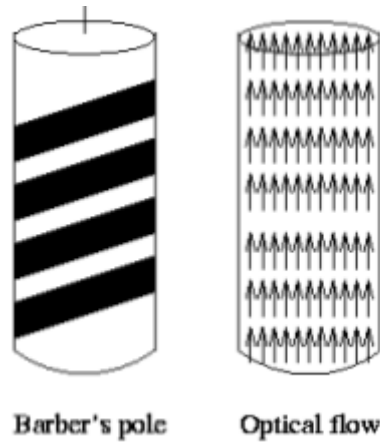


Figure 1. Optical flow of a Barber's pole, adapted from The University of Edinburgh [7].

Optical flow is the pattern of visible motion of objects in an image in between two frames caused by the movement of the object itself or camera [8]. The optical flow is represented as a 2D vector field, each showing the displacement of the movement points from the first frame to the second [7].

The optical flow can be determined by the following equation:

$$f_x u + f_y v + f_t = 0$$

Where f_x and f_y are the image gradients, f_t is the gradient over time, and (u,v) represents the flow vector displacement [9].

The 'unknown' problem (u,v) is also known as the aperture problem where the movement of an object has many possibilities, but it cannot be determined which of the possibilities is a reality, in essence, the motion of an edge as seen through an aperture is ambiguous [7], [10]. However, there are multiple ways of solving the problem.

2.1.1. Lucas-Kanade Method



Figure 2. Neighboring pixels of $N \times N$ pixels, adapted from First Principles of Computer Vision [10].

The Lucas-Kanade method is one of the methods for solving the unknowns. Assuming the flow is locally smooth (object moves consistently or the optical flow is constant), and the neighboring pixels have the same displacement, multiple equations can be made from the neighboring pixels of a $n \times n$ image patch [10]. Doing so makes a system of equations where finding a least squares solution is required [11].

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

A
 25×2

x
 2×1

b
 25×1

Using a 5×5 image patch gives us 25 equations [11].

The LSQ solution involves taking an average of the optical flow estimations within a neighborhood or region, giving us an equation where the unknown variables can be solved algebraically [12].

The Lucas-Kanade algorithm eliminates regions that do not have structure by looking at the invertibility of the matrix S^{TS} through the eigenvalues. This results in a set of optical flow vectors that are distributed over the image/frame, giving an estimation of how each point on the image/frame will move from one frame to another [12].

The main advantage of the Lucas-Kanade method is that the number of operations needed to compute the eigenvalues over time are constant, given that for a neighborhood is of a fixed size. Hence, the complexity of the algorithm is linear in the number of pixels examined for the frame [12].

2.1.2. Horn-Schunck method

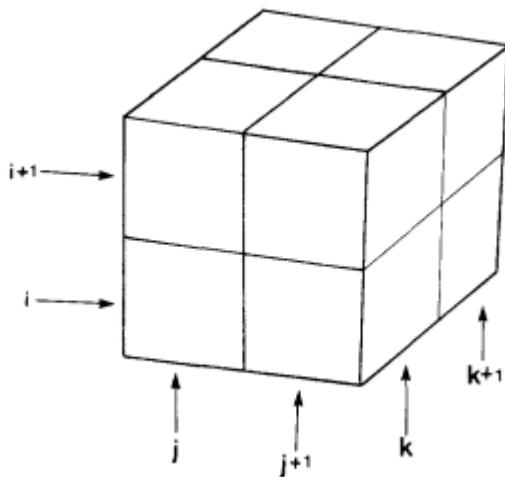


Figure 3. The three partial derivatives of the image brightness, j corresponding to x , i to the y direction, while k is the time direction. Adapted from “Determining Optical Flow” [13].

The Horn-Schunck method is another method for solving the unknowns, where moving objects are initially defined. Assuming that the flow remains smooth over the whole

image, and that the surface of the image is flat, it will try to minimize distortions in flow, by incorporating an extra smoothness constraint expressed to minimize the square of the magnitude of the gradient of the optical flow velocity [13].

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \quad \text{and} \quad \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2.$$

Extra smoothness constraint

Another constraint are the Laplacians of the x and y components of the flow, which can be defined in terms of u and v as:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad \text{and} \quad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}.$$

The Laplacians is determined by the situations, where in simpler situations, both the Laplacians are zero, but if a flat object were to rotate then the second partial derivatives of both u and v will vanish (assuming perspective of the projection in the image formation) [13].

After the constraints have been considered, and using the three partial derivatives of the image's brightness to estimate the Laplacian of the Flow Velocities, a new set of velocity estimate (u^{n+1}, v^{n+1}) can then be derived from the estimated derivatives and the average of the previous velocity estimates (u^n, v^n) [13].

$$\begin{aligned} u^{n+1} &= \bar{u}^n - E_x[E_x \bar{u}^n + E_y \bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2), \\ v^{n+1} &= \bar{v}^n - E_y[E_x \bar{u}^n + E_y \bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2). \end{aligned}$$

The Horn-Schunck method is best when the brightness gradient of the region is not too small, varying in direction from point to point, and when the velocity of the brightness pattern varies smoothly across the entire image [11].

2.2. Deep Learning

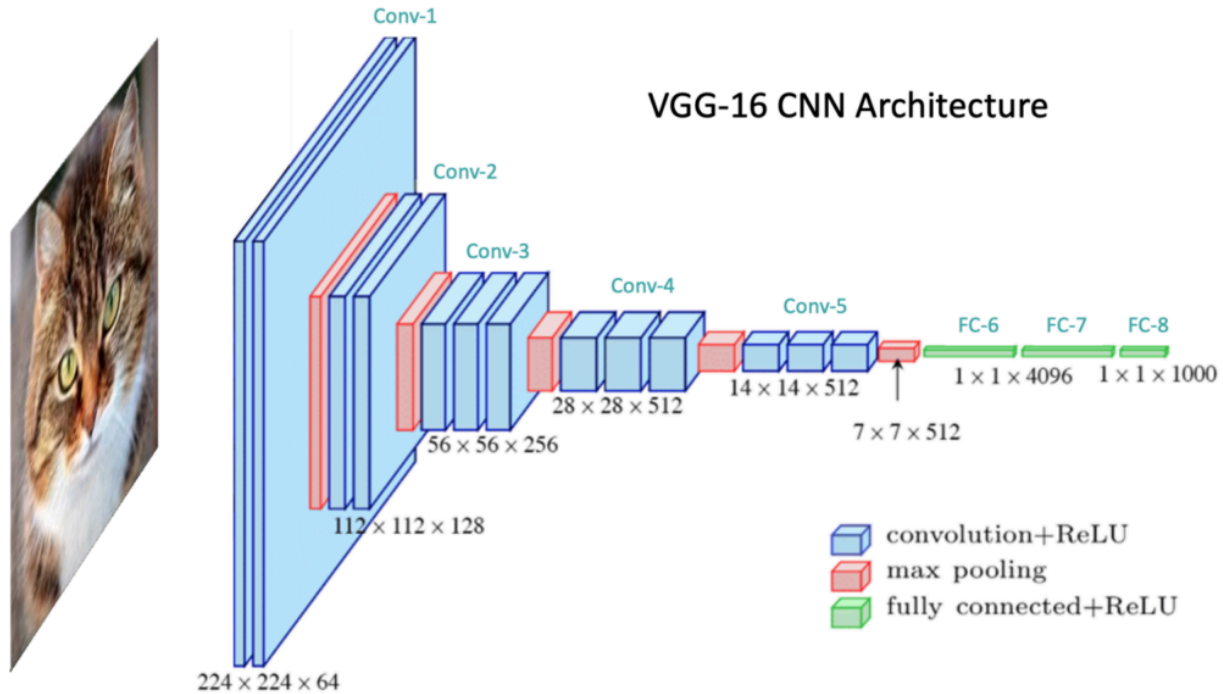


Figure 4. An example for Deep learning's interconnected nodes. Adapted from LearnOpenCV [14].

Deep learning is a subset of machine learning, which contains three or more layers of a neural network. These neural networks are used to simulate how a human brain would function, with each layer having its own function [15]. In the case of frame interpolation, the neural networks will simulate the behavior of the human brain to fill in missing gaps from a low frame rate video, subsequently outputting a synthesized frame in between the frames.

Deep neural networks are mostly used commonly among many frame interpolation programs to calculate the optical flow of the current frame, for example, in DAIN, a pretrained state-of-the-art flow model was used to estimate the optical flow of the frame [16].

In order to maximize the quality of the prediction or outcome, deep neural networks have numerous layers of interconnected nodes that build upon one another. There are

many different types of neural networks each with its own specialization in tackling specific problems, but the most appropriate neural network is CNNs which is used primarily in computer vision [14].

The main reason that CNN is used for computer vision is for its ability in processing that has a grid-like topology, such as an image, or a singular frame [14].

2.3. Convolutional Neural Networks (CNN)

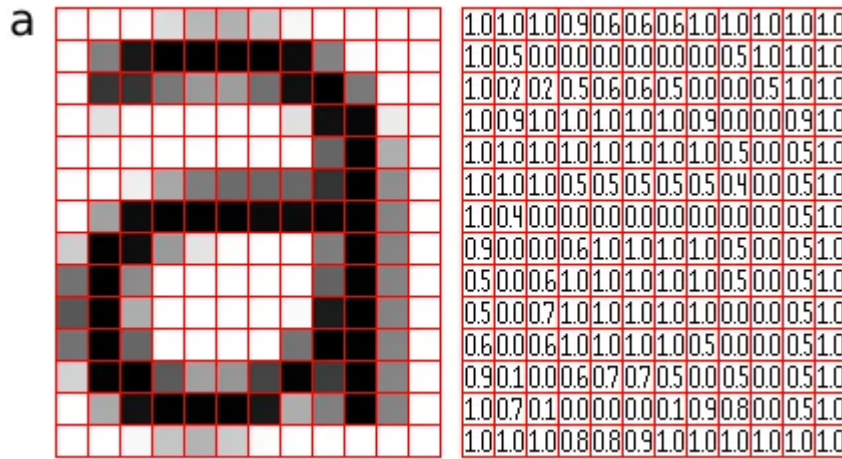


Figure 5. Example for CNN area topology. Adapted from Towards Data Science [17].

CNNs are a class of neural networks that specialize in localized detection, e.g. image patches, or area topology [17]. There are different model architectures which are used to interpolate video frames, the classic model architecture is built on the foundation of the earlier discovered gradient-descent method, and transfer learning [17]. Since then, a number of applications for CNN encodings have been made with that concept, and these include frame interpolation and other computer vision related tasks.

Nicklaus et al. proposes the use of a classic encoder/decoder CNN architecture, where the encoder layer of the network learns a simplified rendering of the original frame,

f0 and f1 [18]. The two frames are encoded and are stacked on top of each other, which allows the decoder network to perform an upscaling algorithm in order to predict how the interpolated frame will move within the context of the two previous encoded frames [17].

All of the CNN based applications (Sepconv, and RIFE) used in this paper builds on the method proposed by Nicklaus et al. above.

3. Methodology

3.1. Testing Data Used

To evaluate the performance of the different types of frame interpolation method and to get data, different videos will be used, with differing types of motion. The different types of motion in the videos will be:

1. Linear motion
2. Rotary motion
3. Oscillatory motion

They will also have a differing amount of speed for each motion [19].

The videos will be limited to 4 fps (frames per second/ frame rate), and the programs will be used to increase it to the industry standard of 24 fps [20].

The effectiveness of the frame interpolation methods will be tested based on the resolutions of the video, 1080p, and 240p. This is done to be able to directly see the side effects of having less data for each of the frame interpolation methods to interpolate, which can help visualize the disadvantages of each frame interpolation method.

3.2. Dependent Variable

3.2.1. Mean Squared Error (MSE)

Mean squared error is used to evaluate the quality of images, or in this case frames [21]. It measures the average error, squared, between two images or frames. MSE can be calculated by:

$$\text{MSE} = \frac{1}{MN} \sum_{n=0}^M \sum_{m=1}^N [\hat{g}(n, m) - g(n, m)]^2$$

MSE is also used to calculate other image quality measurements such as PSNR, and it is used in this paper to provide a quantitative measure of the difference between the synthetic frames and the ground truth frames. Since MSE is a type of regression metrics, the lower the MSE value, the more accurate the interpolated frame is [21].

3.2.2. Structural Similarity Index (SSIM)

Structural Similarity Index Method is also used to evaluate the quality of images, where the image degradation (with factors such as luminance and structural similarity being considered) is calculated [21]. It collaborates with other perception-based models such as visual salience, contrast masking, which analyzes each pixel of a frame [21]. SSIM measures the similarity between the two frames, thus it can be helpful in evaluating the performance of the methods [21].

SSIM is calculated by [21]:

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^\alpha \cdot [c(\mathbf{x}, \mathbf{y})]^\beta \cdot [s(\mathbf{x}, \mathbf{y})]^\gamma$$

The higher the value of SSIM is, the more accurate the structure of the objects in the interpolated frame is.

3.2.3. Peak Signal to Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio is a metric that quantifies the ratio between the maximum attainable signal power and the undesirable noise power that adversely affects the quality of its representation; this ratio is given in decibel form [21].

$$\text{PSNR} = 10\log_{10} (\text{peakval}^2) / \text{MSE}$$

The peakval is the maximum peak value in the image data

The PSNR serves as an approximate measure for assessing the quality of reconstructed content in relation to human perception, as such, higher PSNR value indicates a better quality of the frame.

3.2.4. Learned Perceptual Image Patch Similarity (LPIPS)

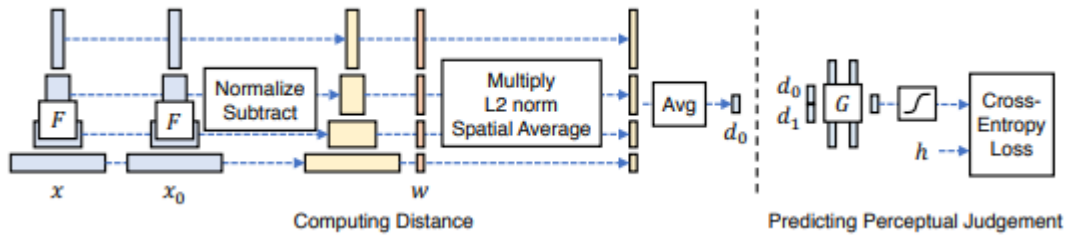


Figure 6. Simplified view of the LPIPS network used in “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric.” [22].

LPIPS is a metric that employs the use of CNN to quantify the perception based on a human's visual perception. It does this by sampling different types of distortion of an image from different manipulated sources such as CNN-based distortions, video deblurring, and resolution and frame interpolation [22].

In addition to those data sets, surveys are done to validate the data sets, and behavior of the convolutional networks [21]. The networks are utilized to evaluate the distance between the two patches of images, and this distance is the averaged and normalized deep embeddings from the spatially and sum channel from network F, which is calculated by:

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l \odot (\hat{y}_{hw}^l - \hat{y}_{0hw}^l)\|_2^2$$

Where x, x_0 is the reference and distorted patches, $\hat{y}^l, \hat{y}_0^l \in \mathbb{R}^{H_l \times W_l \times C_l}$ is the feature stack from L layers, and normalized unit in the channel dimension, and $w_l = 1 \forall l$ to compute the cosine distance [21].

The value is then scaled by the vector w , taking the distance of l_2 from the F network [21]. Finally, it is averaged across all spatial dimensions and layers, giving the distance d_0 [21]. This is passed to a small network to predict the LPIPS from the distance pair of d_0 , and d_1 [22]. This whole process can be visualized from Figure 4, showing the layers of the CNN.

The small network used in this research is the AlexNet network, which is more closely matched to the architecture of the human visual cortex [22]. The lower the LPIPS score, the better as the images are perceptually similar [22].

4. Results

4.1. Linear Motion

4.1.1. Tabular data representation

Below are the tables showing the different means of the different metrics used, across all the interpolated frames of the programs used. The format used will be the same across all of the different types of motion.

Blue: CNN methods

Red: Optical flow methods

	MSE	SSIM	PSNR	LPIPS
RIFE	42.3619	0.9883	3.7879	0.0435
SepConv	39.8707	0.9851	6.7099	0.0494
Butterflow	40.7356	0.9879	5.5060	0.0434
Softmax	38.1894	0.9848	10.0446	0.0493
CNN	41.1163	0.9867	5.2489	0.0465
Optical flow	39.4625	0.9864	7.7753	0.0464

Table 1. Linear fast speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	40.4414	0.9757	5.9748	0.0338
SepConv	36.8067	0.9700	13.6405	0.0388
Butterflow	38.1069	0.9736	10.1225	0.0349
Softmax	36.3394	0.9703	15.2509	0.0377
CNN	38.6241	0.9729	9.8077	0.0363
Optical flow	37.2232	0.9720	12.6867	0.0363

Table 2. Linear fast speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	35.4954	0.8986	25.4461	0.1594
SepConv	33.7380	0.9008	32.6882	0.1850
Butterflow	34.9352	0.9208	27.9772	0.1520
Softmax	33.5939	0.9006	33.3878	0.1799
CNN	34.6167	0.8997	29.0672	0.1722
Optical flow	34.2646	0.9107	30.6825	0.1660

Table 3. Linear medium speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	34.4386	0.8465	29.0309	0.1415
SepConv	33.1019	0.8156	36.0975	0.1749
Butterflow	33.6012	0.8469	33.7413	0.1487
Softmax	33.0942	0.8306	35.7277	0.1612
CNN	33.7703	0.8312	32.5642	0.1582
Optical flow	33.3477	0.8388	34.7345	0.1550

Table 4. Linear medium speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	32.3899	0.7325	37.5756	0.1935
SepConv	31.3394	0.6838	48.6020	0.2664
Butterflow	32.3957	0.7638	38.3058	0.1802
Softmax	31.0622	0.6762	51.9549	0.2649
CNN	31.8647	0.7082	43.0888	0.2300
Optical flow	31.7290	0.7200	45.1304	0.2226

Table 5. Linear slow speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	32.4792	0.7427	36.8736	0.1138
SepConv	31.4210	0.6472	47.4847	0.1835
Butterflow	32.2167	0.7593	39.2660	0.1254
Softmax	31.1858	0.6459	50.2775	0.1810
CNN	31.9501	0.6950	42.1792	0.1487
Optical flow	31.7013	0.7026	44.7718	0.1532

Table 6. Linear slow speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	36.2677	0.8641	23.1148	0.1143
SepConv	34.3796	0.8338	30.8705	0.1500
Butterflow	35.3319	0.8754	25.8198	0.1141
Softmax	33.9108	0.8347	32.7739	0.1457
CNN	35.3237	0.8489	26.9926	0.1320
Optical flow	34.6214	0.8551	29.2969	0.1299

Table 7. Average linear results

4.1.2. Visual representation

The metrics to be visualized in this paper will be PSNR as it measures the error between the groundtruth and the interpolated frame, in terms of a ratio between original data and the error yielded by the interpolation. This value can be the approximate estimation to human perception, similar to LPIPS, of the interpolation quality, hence a higher value will represent a better result. This will apply to all different types of motion.

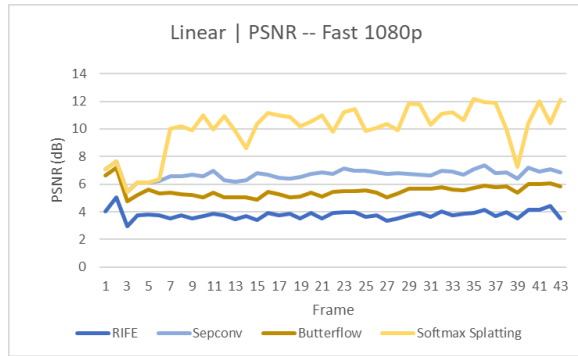


Figure 7. Linear fast speed 1080p PSNR results

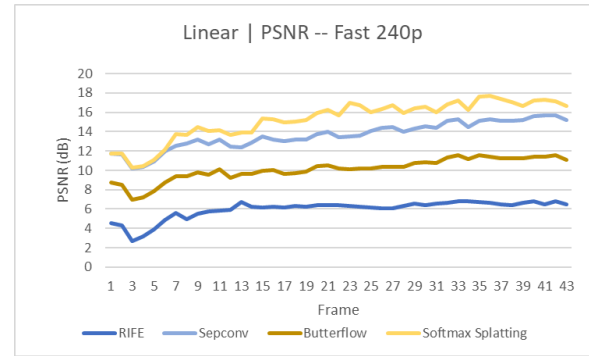


Figure 10. Linear fast speed 240p PSNR results

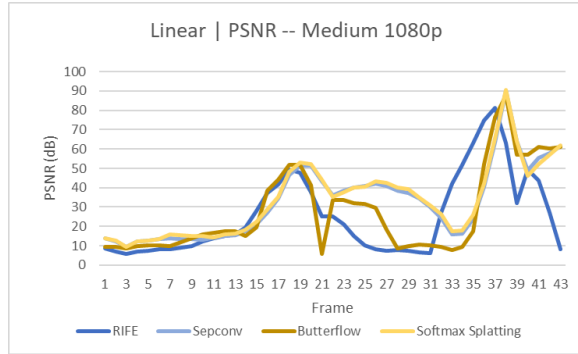


Figure 8. Linear medium speed 1080p PSNR results

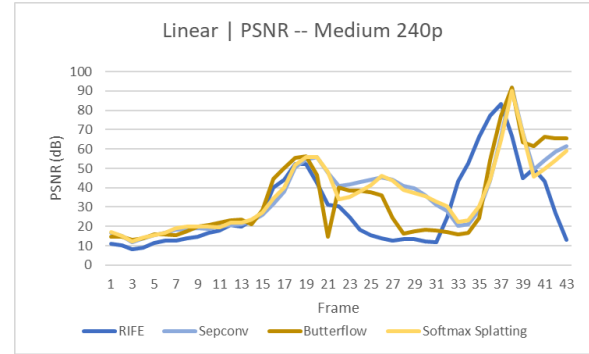


Figure 11. Linear medium speed 240p PSNR results

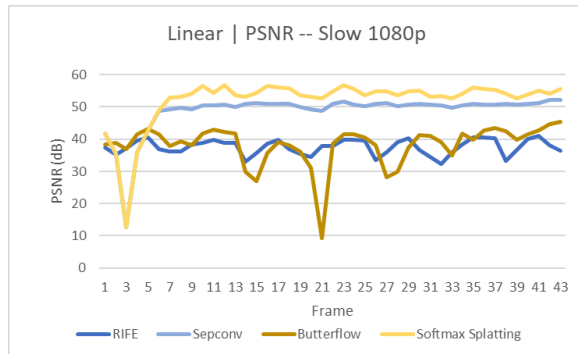


Figure 9. Linear slow speed 1080p PSNR results

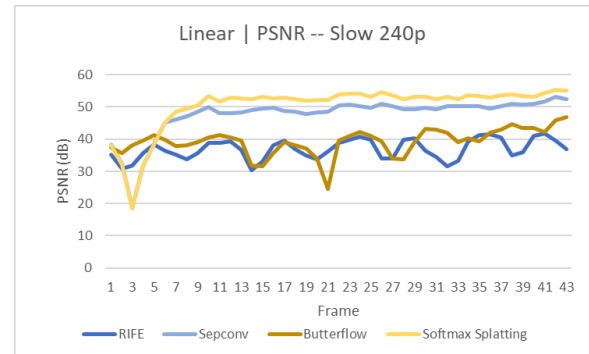


Figure 12. Linear slow speed 240p PSNR results

4.1.3. Data analysis

From the graphs above, it can be inferred that optical flow based methods consistently outperform CNN based methods. However, the extent of the outperformance for optical flow based methods decreases with speed and resolution.



Figure 13. Comparison between programs frame 22 of 1080p Linear slow speed.

Generally, optical flow methods have a better result than CNN based methods, with optical flow having less MSE, LPIPS, and more SSIM and PSNR. On videos with objects that have a lower speed, both methods perform about the same, with the MSE of both of the methods ranging around the same value (CNN = 31.8647, Optical flow = 31.7290).

CNN also has more distortion on lower resolution, which can be visualized from the LPIPS value of optical flow having a lower overall average (0.1359) compared to CNN based methods (0.1611). This distortion can be visualized from the pictures above, where CNN methods have more visual artifacts compared to optical flow methods.



Figure 14. Comparison between programs frame 22 of 240p Linear fast speed.

Similarly, on lower quality, CNN method's frames compression artifacts are less visible compared to optical flow's frames. This effect is shown in the frames below. Optical flow methods produced a higher concentration of differences shown in gray compared to CNN methods.

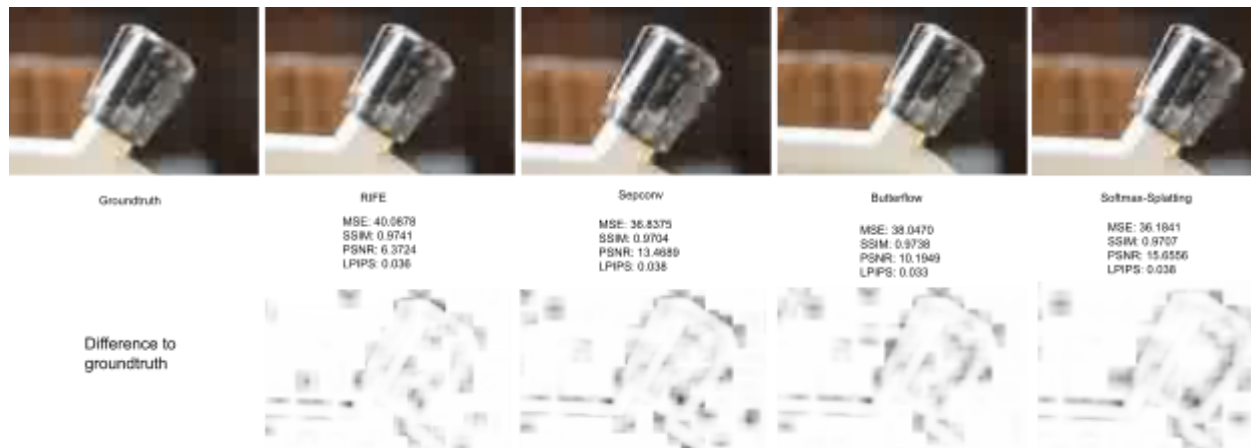


Figure 15. Zoomed in comparison between programs frame 22 of 240p Linear fast speed, with difference mask.

Optical flow's PSNR also always stays above the 30 dB range for most of the test cases, which is deemed as the acceptable value for evaluating images [23]. This indicates that optical flow methods have a higher quality of interpolated frames, which is also backed up by the lower LPIPS and MSE.

Overall, although both methods suffer from data loss at the fastest speed, optical flow appears to perform better at higher speed. The average of optical flow's MSE(38.3428) is lesser compared to CNN's MSE (39.8702) on faster speed, and this effect can be seen from the water drop where CNN appears to struggle to render the frames beyond f1 of the groundtruth (figure 6).

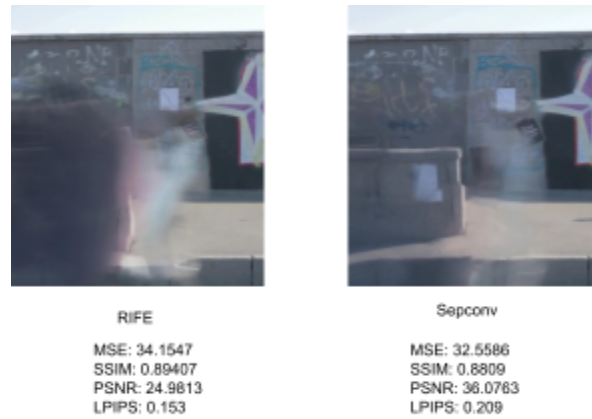


Figure 16. Comparison between CNN programs frame 39 of 1080p Linear fast speed.

However, it is to be noted that both techniques follow the same trend when it comes to the decrease of MSE with the speed of the objects. The decrease of MSE, from fast speed to slow speed, of the CNN based methods (1080p = 9.2517, 240p = 6.7395) is greater compared to Optical flow methods (1080p = 7.7336, 240p = 5.5219), which indicates that CNN performs dramatically better on lower linear speed, compared to optical flow methods.

While Optical flow has a greater average on the objective measurements, CNN provides less artifacts, but also has more distortion effect, seen in the images above. The average SSIM value of Optical flow(0.8551) and CNN (0.8489) indicates that both methods have around the same performance, with only a 0.0062 difference.

4.2. Rotary Motion

4.2.1. Tabular data representation

	MSE	SSIM	PSNR	LPIPS
RIFE	32.8434	0.8628	34.8179	0.1927
SepConv	31.7671	0.8665	43.7671	0.2109
Butterflow	32.8615	0.8657	35.7873	0.1982
Softmax	31.5022	0.8472	46.3804	0.2170
CNN	32.3053	0.8647	39.2925	0.2018
Optical flow	32.1819	0.8565	41.0839	0.2076

Table 8. Rotary fast speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	33.3961	0.8575	30.8582	0.1659
SepConv	32.5909	0.8313	36.9249	0.1851
Butterflow	33.2991	0.8687	32.6129	0.1634
Softmax	32.1547	0.8287	40.9898	0.1885
CNN	32.9935	0.8444	33.8916	0.1755
Optical flow	32.7269	0.8487	36.8014	0.1760

Table 9. Rotary fast speed 240p results'

	MSE	SSIM	PSNR	LPIPS
RIFE	32.0448	0.9468	40.9124	0.1909
SepConv	31.3577	0.9489	47.6376	0.2067
Butterflow	32.6006	0.9582	38.2344	0.1687
Softmax	31.2133	0.9475	49.2492	0.2081
CNN	31.7013	0.9479	44.2750	0.1988
Optical flow	31.9070	0.9529	43.7418	0.1884

Table 10. Rotary medium speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	32.6109	0.8558	36.8898	0.1803
SepConv	31.8894	0.8378	43.0072	0.1969
Butterflow	32.4945	0.8670	38.5514	0.1747
Softmax	31.8309	0.8400	43.6547	0.1917
CNN	32.2502	0.8468	39.9485	0.1886
Optical flow	32.1627	0.8535	41.1031	0.1832

Table 11. Rotary medium speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	34.4014	0.9274	23.6654	0.0493
SepConv	32.9836	0.8760	32.8078	0.0637
Butterflow	34.4954	0.9369	23.2239	0.0452
Softmax	32.8123	0.8751	34.1670	0.0638
CNN	33.6925	0.9017	28.2366	0.0565
Optical flow	33.6539	0.9060	28.6955	0.0545

Table 12. Rotary slow speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	34.2121	0.9419	24.7795	0.0433
SepConv	32.7064	0.9092	35.0444	0.0521
Butterflow	33.2889	0.9309	30.5825	0.0569
Softmax	32.6292	0.9107	35.7028	0.0521
CNN	33.4593	0.9256	29.9120	0.0477
Optical flow	32.9591	0.9208	33.1427	0.0545

Table 13. Rotary slow speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	33.2515	0.8987	31.9872	0.1371
SepConv	32.2159	0.8783	39.8648	0.1526
Butterflow	33.1733	0.9046	33.1654	0.1345
Softmax	32.0238	0.8749	41.6907	0.1535
CNN	32.7337	0.8885	35.9260	0.1448
Optical flow	32.5986	0.8897	37.4280	0.1440

Table 14. Average rotary results

4.2.2. Visual representation

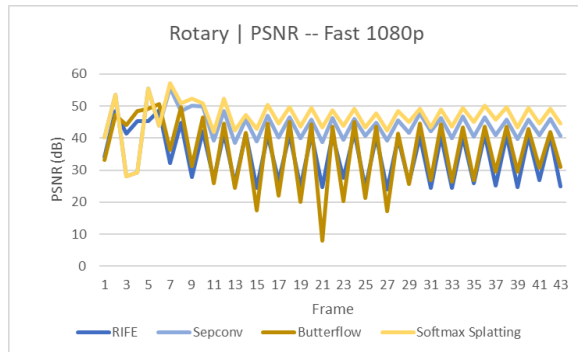


Figure 17. Rotary fast speed 1080p PSNR results

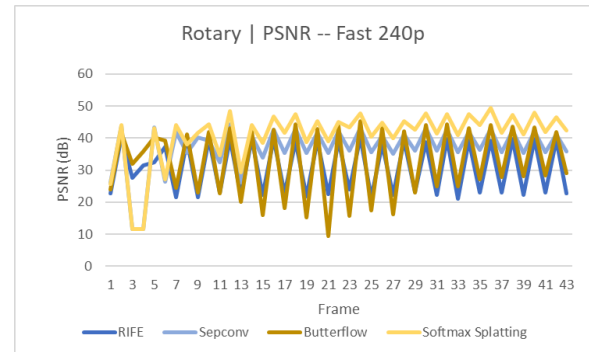


Figure 20. Rotary fast speed 240p PSNR results

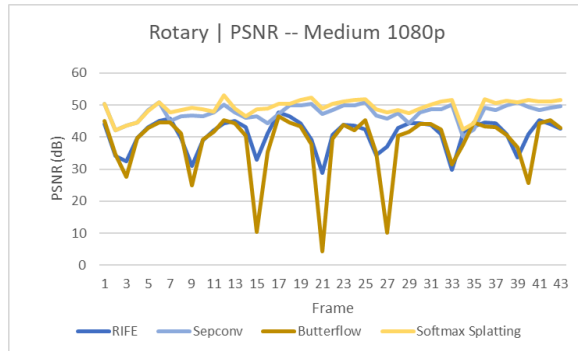


Figure 18. Rotary medium speed 1080p PSNR results

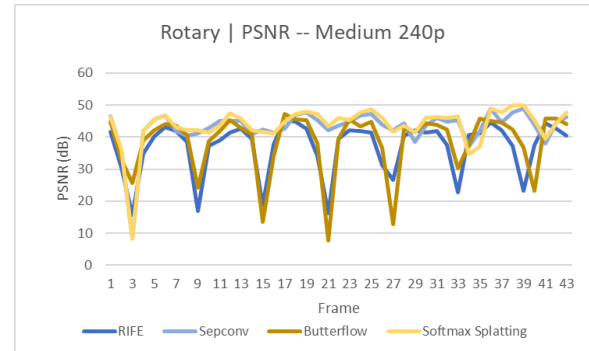


Figure 21. Rotary medium speed 240p PSNR results

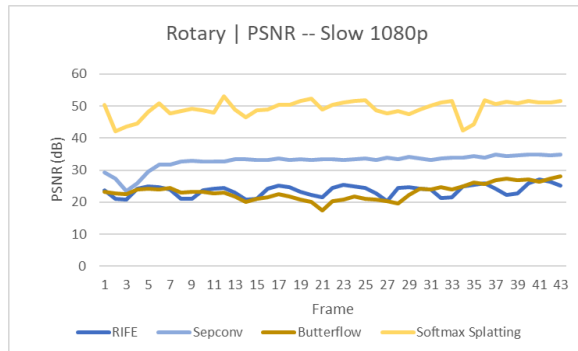


Figure 19. Rotary slow speed 1080p PSNR results

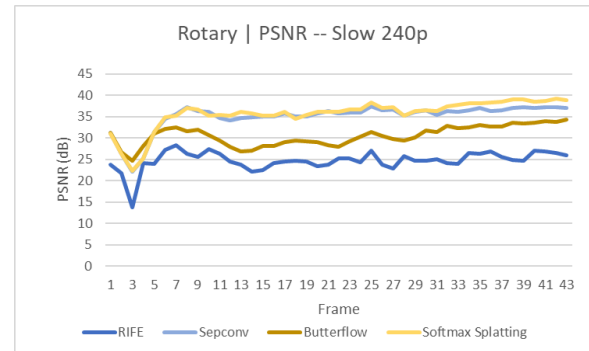


Figure 22. Rotary slow speed 240p PSNR results

4.2.3. Data analysis

The graphs indicate that for rotary motion, there is not that much of a difference. The difference only increased with slower motion, where the graph of optical flow performed slightly better, indicated by the higher position.

Both CNN and optical flow based interpolation methods yield a similar result in terms of SSIM, with the SSIM of CNN methods averaging at 0.8885, and optical flow based methods averaging at 0.8897. This reveals that in terms of the reconstruction of the objects in the frames, both are similar, with CNN only being better by 0.0012.

Similarly, the LPIPS of CNN based methods averages at 0.1448 while optical flow based methods averages at 0.1440, with optical flow based methods performing better by 0.55%. To the human eye, frames from both methods are visually the same.

While the graph may indicate that there is no difference between the PSNR of CNN methods, and optical flow methods, the averages between them says otherwise, with CNN based methods at 35.9260 dB and optical flow methods with 37.4281 dB. Yet, despite the differences, CNN still produces a clearer output. This is shown in figure 9, where optical flow methods have a residue from the previous frames.

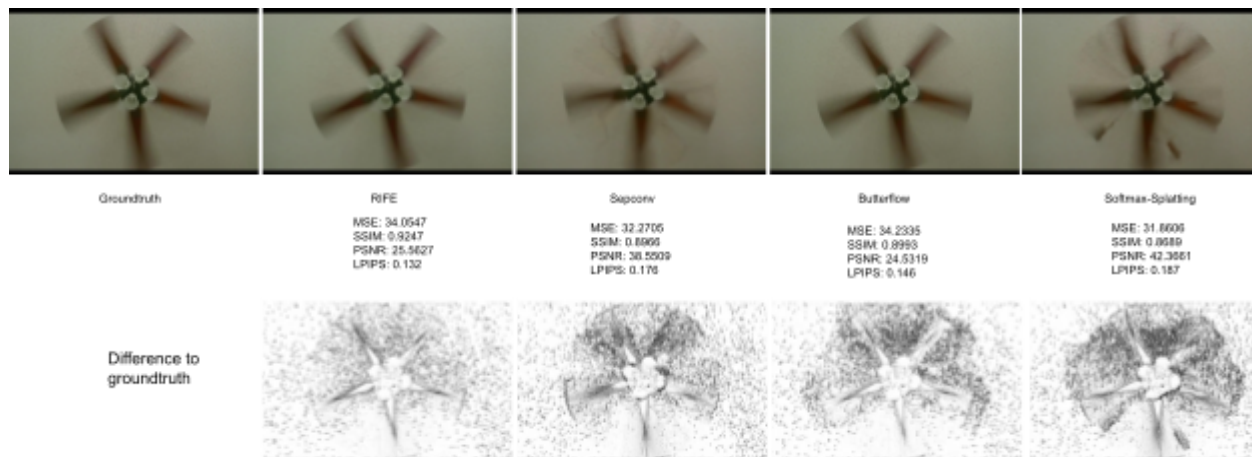


Figure 23. Comparison between programs on frame 13 of 1080p rotary fast speed, with difference mask

However, unlike linear motion, both methods perform better on lower resolution, evidenced by the frames having less artifacts. This is also shown by the differences of PSNR between the two motions, with rotary motion having a higher average PSNR (CNN = 35.9260dB, Optical flow = 37.4281 dB), while linear motion has a lower average PSNR (CNN = 26.9930 dB, Optical flow = 29.2969 dB).

Although rotary motion has less distortion and more image fidelity, it still displays some inconsistencies, particularly at darker areas.



Figure 24. Zoomed in comparison between programs on frame 32 of 240p rotary slow speed

When it comes to the MSE, rotary motion displays no clear trend unlike linear motion, indicating that the frame interpolation quality does not depend on the speed of the objects in the image. However, compared to linear motion, rotary motion averages a lower MSE (CNN = 32.7337, Optical flow = 32.5152), while linear motion averages a higher MSE (CNN = 35.3237, Optical flow = 34.6214). This evinces that frame interpolation methods work better on rotary motion. Furthermore, CNN methods have a higher MSE compared to optical flow, which signals that optical flow methods perform better in terms of errors.

Despite both methods performing similarly on the metrics, CNN methods again show a lower rate of visual distortion and higher fidelity, seen in figure 9 and 10. However, optical flow methods perform better in terms of objective metrics.

4.3. Oscillatory Motion

4.3.1. Tabular data representation

	MSE	SSIM	PSNR	LPIPS
RIFE	35.7826	0.9455	17.8797	0.0807
SepConv	34.4231	0.9194	23.8847	0.0974
Butterflow	35.1683	0.9353	20.5715	0.0877
Softmax	34.6143	0.9258	22.8976	0.0908
CNN	35.1029	0.9325	20.8822	0.0891
Optical flow	34.8913	0.9306	21.7346	0.0893

Table 15. Oscillatory fast speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	35.9656	0.9268	17.0147	0.1050
SepConv	34.3615	0.9044	24.0841	0.1217
Butterflow	34.8435	0.9101	21.7651	0.1212
Softmax	34.4581	0.9129	23.6078	0.1036
CNN	35.1636	0.9156	20.5494	0.1134
Optical flow	34.6508	0.9115	22.6865	0.1124

Table 16. Oscillatory fast speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	35.7444	0.9120	25.1582	0.1583
SepConv	37.8800	0.9212	10.9506	0.0731
Butterflow	39.1203	0.9488	8.1177	0.0614
Softmax	38.2662	0.9379	9.9631	0.0665
CNN	36.8122	0.9166	18.0544	0.1157
Optical flow	38.6933	0.9434	9.0404	0.0640

Table 17. Oscillatory medium speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	37.4731	0.9285	12.0207	0.0584
SepConv	35.8662	0.8957	17.3734	0.0843
Butterflow	36.7331	0.9274	14.1158	0.0665
Softmax	35.8910	0.9028	17.2270	0.0801
CNN	36.6697	0.9121	14.6971	0.0714
Optical flow	36.3121	0.9151	15.6714	0.0733

Table 18. Oscillatory medium speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	35.1482	0.8932	20.1025	0.1154
SepConv	33.2044	0.8783	31.5131	0.1453
Butterflow	35.2121	0.9038	20.0738	0.1009
Softmax	32.7189	0.8795	35.4086	0.1450
CNN	34.1763	0.8858	25.8078	0.1304
Optical flow	33.9655	0.8917	27.7412	0.1230

Table 19. Oscillatory slow speed 1080p results

	MSE	SSIM	PSNR	LPIPS
RIFE	34.0440	0.8484	25.9470	0.0824
SepConv	32.7472	0.8158	35.0749	0.1120
Butterflow	33.3703	0.8538	30.2092	0.0900
Softmax	32.3649	0.8164	38.5243	0.1112
CNN	33.3956	0.8321	30.5110	0.0972
Optical flow	32.8676	0.8351	34.3668	0.1006

Table 20. Oscillatory slow speed 240p results

	MSE	SSIM	PSNR	LPIPS
RIFE	35.6930	0.9091	19.6871	0.1000
SepConv	34.7471	0.8891	23.8135	0.1056
Butterflow	35.7413	0.9132	19.1422	0.0880
Softmax	34.7189	0.8959	24.6047	0.0995
CNN	35.2200	0.8991	21.7503	0.1028
Optical flow	35.2301	0.9045	21.8735	0.0937

Table 21. Average oscillatory results

4.3.2. Visual representation

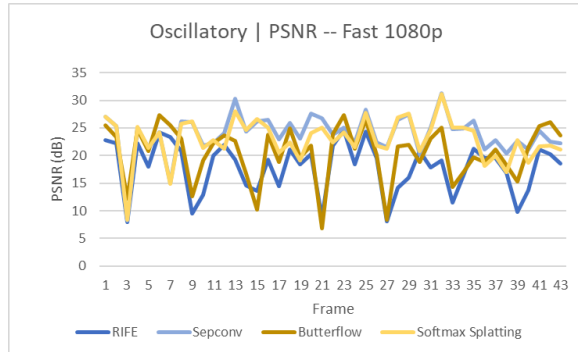


Figure 25. Oscillatory fast speed 1080p PSNR results

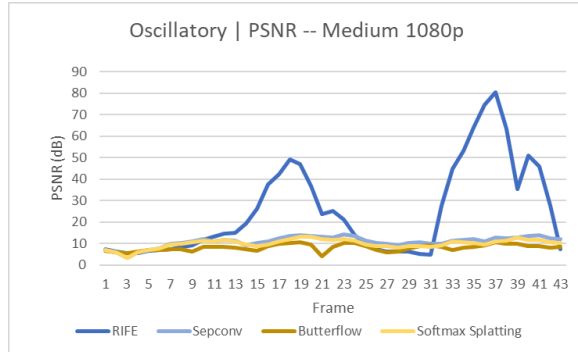


Figure 26. Oscillatory medium speed 1080p PSNR results

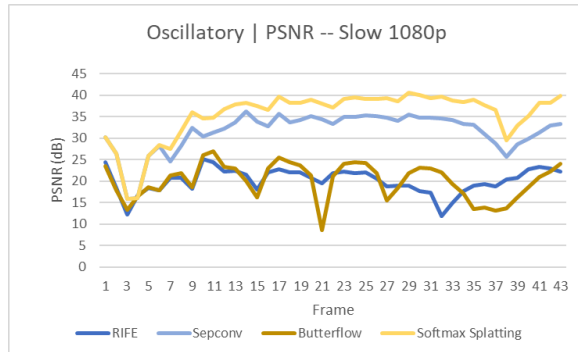


Figure 27. Oscillatory slow speed 1080p PSNR results

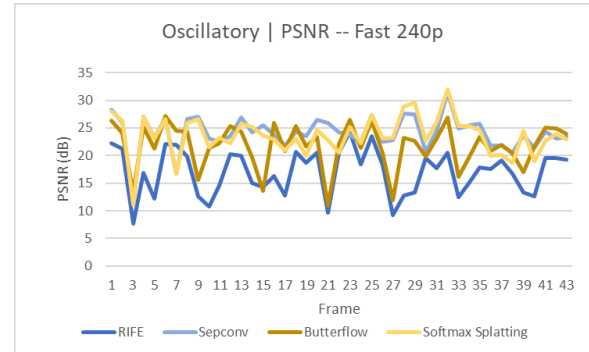


Figure 28. Oscillatory fast speed 240p PSNR results

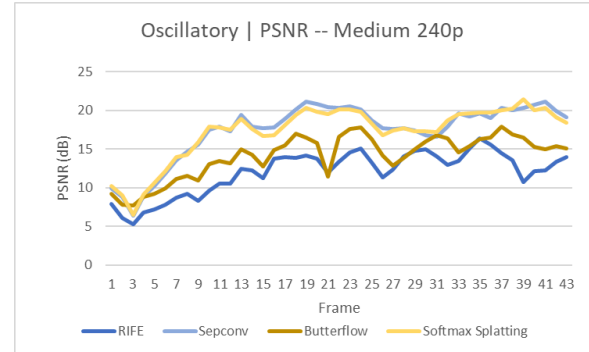


Figure 29. Oscillatory medium speed 240p PSNR results

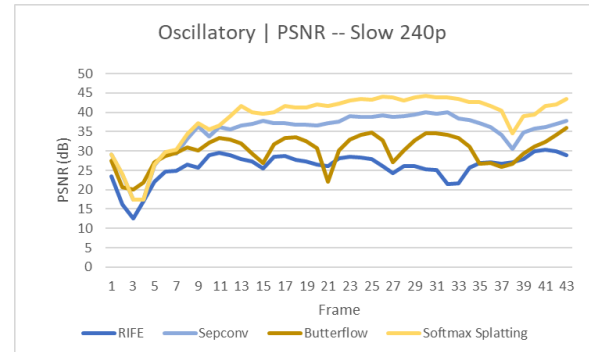


Figure 30. Oscillatory slow speed 240p PSNR result

4.3.3. Data analysis

The graph shows a trend putting optical flow in favor. At faster speeds, there is barely any difference between the PSNR of CNN methods and optical flow methods, except for medium speed where RIFE shows a spike at certain times. At lower speeds however, optical flow methods can be seen outperforming CNN methods, but only to a medium extent.

In most cases, optical flow methods perform better than CNN methods, with optical flow methods outperforming CNN methods frequently in all of the metrics. However, CNN methods perform in medium speed and 1080p resolution, with CNN having higher PSNR (CNN = 18.0544 dB, Optical flow = 9.0404 dB) and lower MSE (CNN = 36.8122, Optical flow = 38.6933), while, in terms of LPIPS (CNN = 0.1157, Optical flow = 0.0640) and SSIM (CNN = 0.9166, Optical flow = 0.9434), they are perform worse.

This means that CNN methods have less accuracy when it comes to recreating the structure of the original frames, f0 and f1. This can be seen in the frames below where a ball can be seen to have disappeared completely, or have blended into the background.

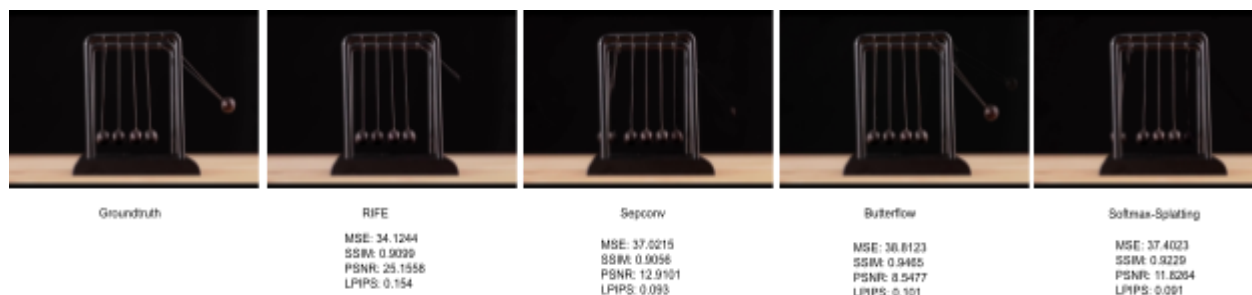


Figure 31. Comparison between programs on frame 22 of 1080p oscillatory medium speed

In terms of MSE as a whole, CNN methods and optical flow methods appear to have minimal differences, with CNN methods averaging 35.2200, and optical flow methods averaging at 35.2301, meaning that in terms of interpolation errors, both perform similarly, with optical flow methods performing better only by 0.101.

However, in terms of LPIPS, optical flow methods have a lower average at 0.0937, compared to CNN methods at 0.1028. This error can be visualized in the frames below where CNN appears to have problems rendering the sliding weight of the metronome, as it stretches the base.

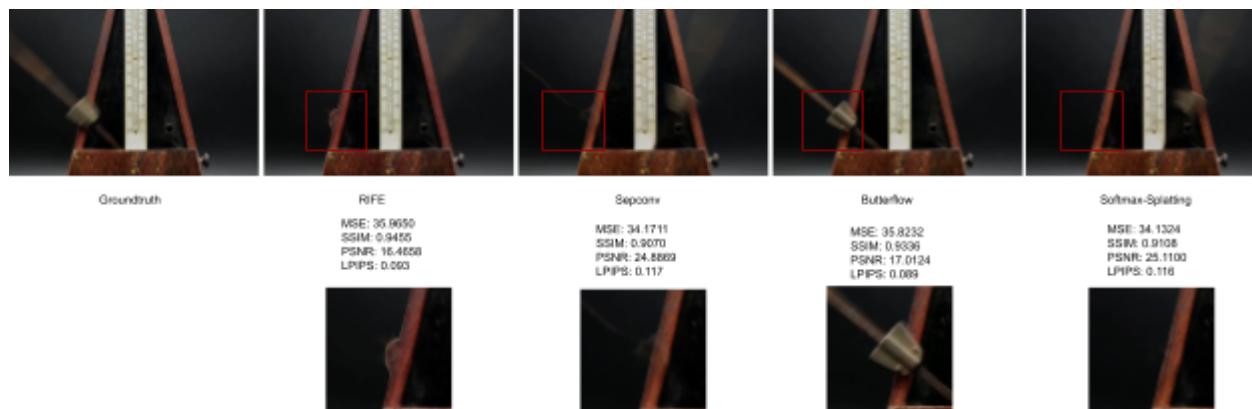


Figure 32. Zoomed in comparison between programs on frame 34 of 1080p oscillatory fast speed

CNN methods also appear to have more visual distortion or artifacts as shown below, where some parts of the previous frame are left in the current frame.

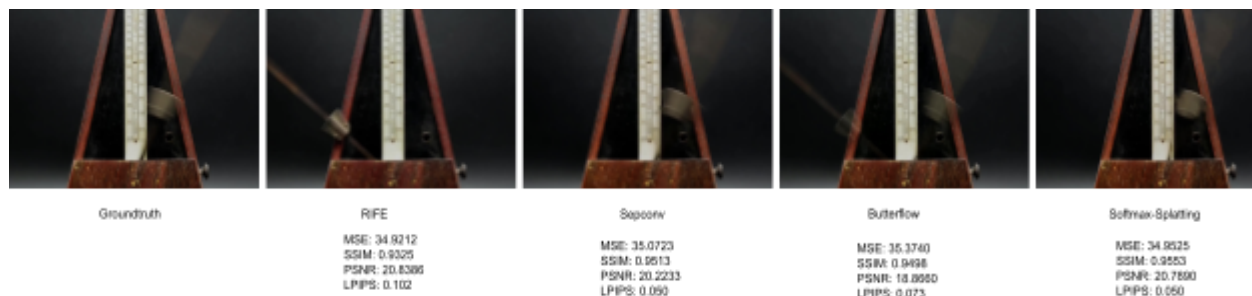


Figure 33. Comparison between programs on frame 30 of 1080p oscillatory fast speed

Unlike the other motions, there is a difference between the SSIM of CNN methods and optical flow methods, with CNN methods performing worse at an average of 0.8991, while optical flow methods performed better at 0.9045. This can be visualized by figure 12, showing the differences between the groundtruth and interpolated frames, where CNN methods can be seen having more gray spots.



Figure 34. Comparison between programs on frame 13 of 1080p rotary fast speed, with difference mask

The same thing can be seen in faster speed, where optical flow methods show a better color accuracy in some parts of the frame (the seat).

The results of oscillatory motion show the opposite of other motion, with CNN methods having more visual distortion and artifacts, and both methods having different SSIM, with CNN methods performing worse. Although Optical flow methods have a lower average LPIPS, CNN most of the time have a lower LPIPS, when compared individually indicating that in most cases, CNN will look the most realistic.

5. Overall results

Overall, in terms of objectivity, optical flow performs way better than CNN. However, in terms of visual clarity, CNN outperforms optical flow. Especially as optical flow has a visual artifact, where two frames are blended together making the subject transparent in some cases (Figure 13).

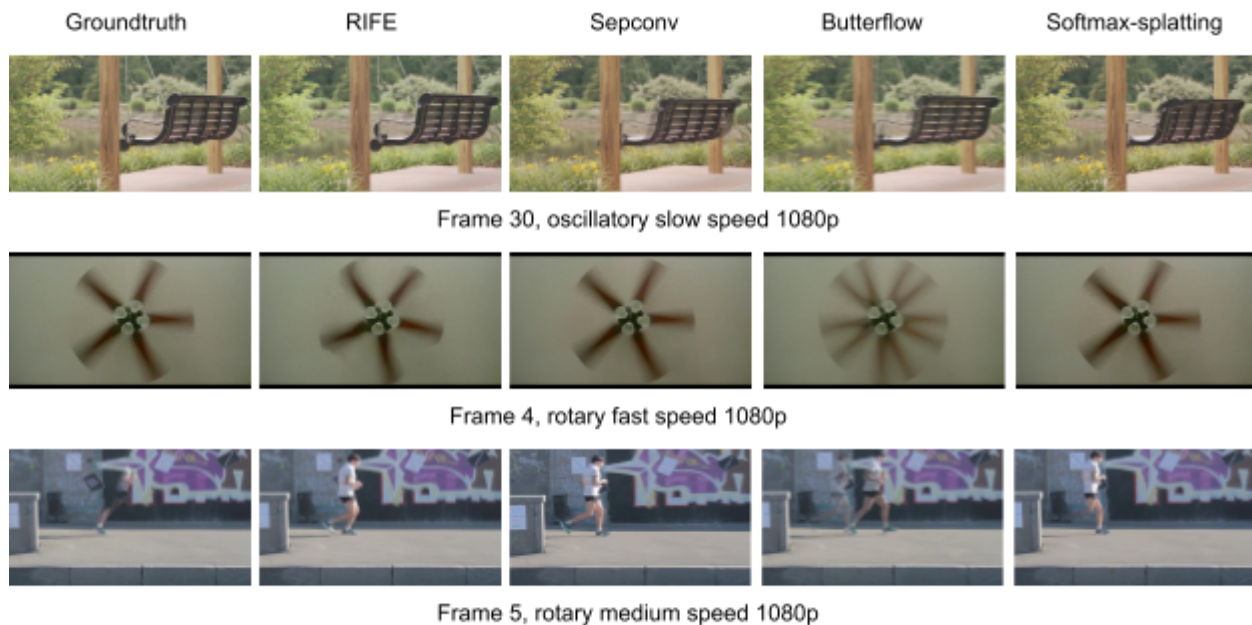


Figure 35. Comparison between programs, showing the transparency effect on the objects

However, this is also a flaw for CNN as at faster speeds, objects are not rendered at all (Figure 14). It is also noticed that both performed exceptionally good in terms of visual artifact when it comes to slower speed with repeated pattern, (such as rotary, oscillatory and sometimes linear motion)

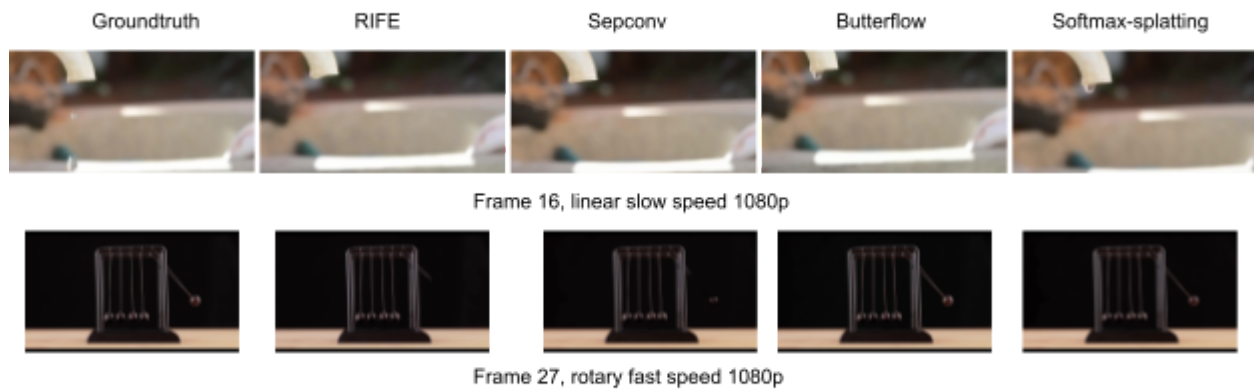


Figure 36. Comparison between programs, showing the disappearance of objects

Another visual artifact is the mosaic effect, which is apparent on lower resolutions. On lower resolutions, there is a trend in which the frames will have a higher noise or a larger amount of checkerboard artifacts as the interpolated frame goes farther than the initial frame (f_0) (Figure 15).

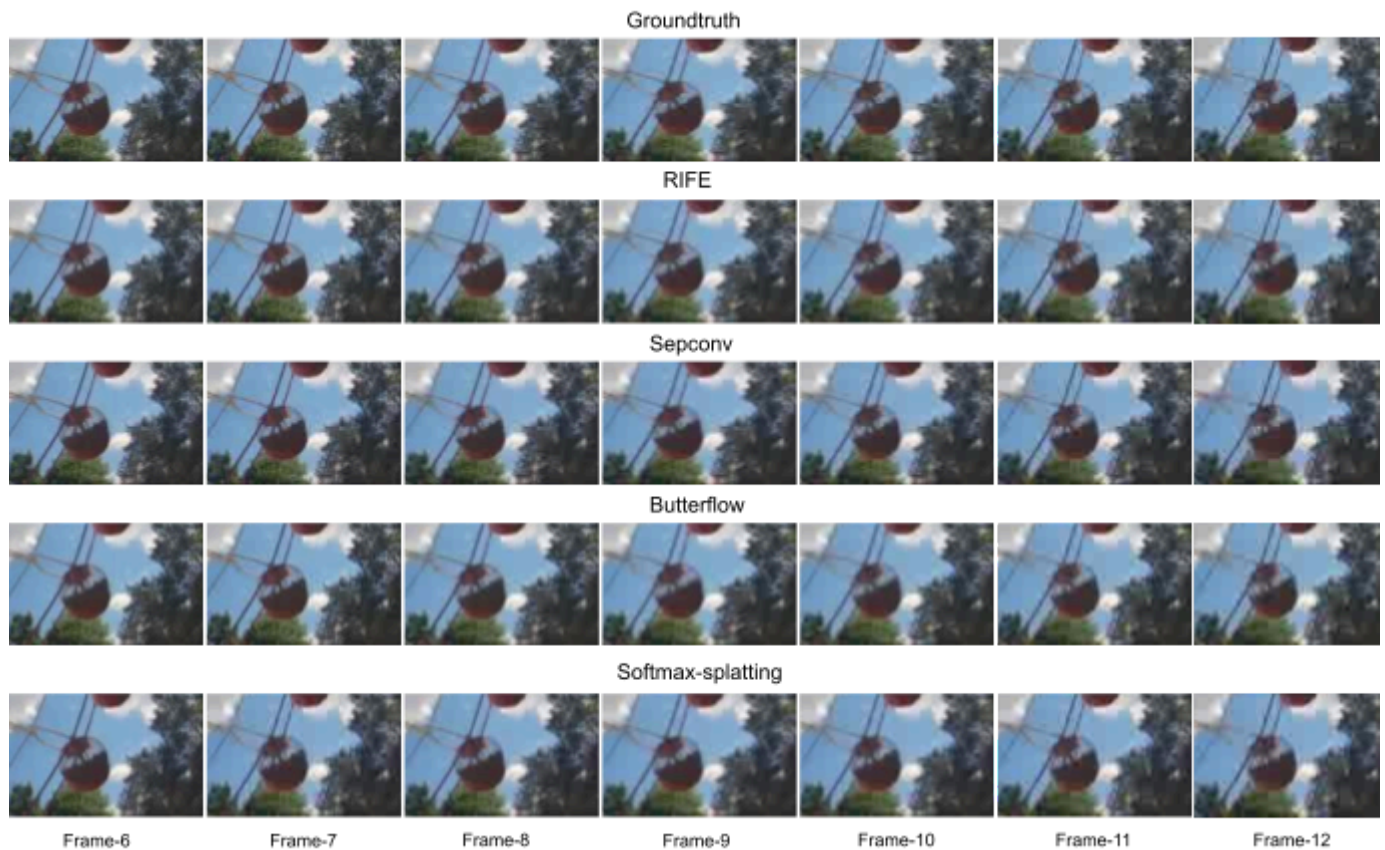


Figure 37. Zoomed in comparison between programs from frame 6 to 12, frame 6 to 12 are used since the checkerboard progression is more apparent.

On multiple metrics, optical flow consistently performs better than CNN, however, for PSNR, CNN generally performs better, indicating that CNN methods have a better performance when it comes to the reconstruction of the frames (less image noise). This is evident with CNN interpolated frames having less defined checkerboard artifacts (Figure 15).

	1080p				240p			
	MSE	SSIM	PSNR	LPIPS	MSE	SSIM	PSNR	LPIPS
RIFE	35.1291	0.9008	25.4829	0.1315	35.0068	0.8804	24.3962	0.1027
Sepconv	34.0627	0.8867	30.9512	0.1442	33.4990	0.8474	32.0813	0.1277
Butterflow	35.2805	0.9135	38.2062	0.1153	34.2171	0.8820	27.8852	0.1091
Softmax	33.7747	0.8962	32.6059	0.1428	33.3276	0.8509	33.4028	0.1230

Table 22. Average program results for 1080p and 240p

As for resolution, optical flow methods perform better in regards to the lowering of resolution as the difference between the values are higher (Table 19).

In terms of motions, optical flow seems to be the better performing method, as it outperforms CNN in most scenarios, whereas CNN methods are only comparable on some scenarios such as on Linear medium 240p, where the differences between optical flow and CNN method metrics is not that large, shown on the tables below.

	Fast				Medium				Slow			
	MSE	SSIM	PSNR	LPIPS	MSE	SSIM	PSNR	LPIPS	MSE	SSIM	PSNR	LPIPS
RIFE	36.7985	0.9277	18.3889	0.1036	34.6345	0.8980	28.2430	0.1481	33.7791	0.8477	20.0831	0.0996
Sepconv	34.9700	0.9128	24.8352	0.1172	33.9672	0.8867	31.2924	0.1525	32.4003	0.8017	38.4212	0.1372
Butterflow	35.8358	0.9236	15.6254	0.1081	34.9142	0.9115	24.2351	0.1287	33.4965	0.8581	30.2769	0.0998
Softmax	34.5430	0.9116	26.5285	0.1145	33.9766	0.8932	31.5349	0.1479	32.1289	0.8006	41.0059	0.1363

Table 23. Comparison between program results, for different speeds

	Fast				Medium				Slow			
	MSE	SSIM	PSNR	LPIPS	MSE	SSIM	PSNR	LPIPS	MSE	SSIM	PSNR	LPIPS
RIFE	-2.164	-0.0297	+9.8541	+0.0445	34.6345	0.8980	28.2430	0.1481	+0.8554	+0.0503	+8.1599	+0.0485
Sepconv	-1.0028	-0.0261	+6.4572	+0.0353	33.9672	0.8867	31.2924	0.1525	+1.5669	+0.0850	-7.1288	+0.0153
Butterflow	-0.9216	-0.0121	+8.6097	+0.0206	34.9142	0.9115	24.2351	0.1287	+1.4177	+0.0534	-6.0418	+0.0289
Softmax	-0.5664	-0.0184	+5.0064	+0.0334	33.9766	0.8932	31.5349	0.1479	+1.8477	+0.0926	-9.4710	+0.0116

Table 24. Difference comparison between program results, for different speeds

6. Conclusion

In this paper, two methods, CNN and optical flow, are analyzed, showing the performance of each method on different metrics such as MSE, SSIM, PSNR, LPIPS. In all other metrics, optical flow seems to be the better performing method, outperforming CNN in MSE, PSNR, and LPIPS. However, the SSIM remains consistent showing that in terms of structural similarity within the interpolated frames, both methods are similar.

While optical flow may perform better objectively, CNN methods have better results in image clarity, especially as CNN methods consistently have lesser artifacts, and do not have the transparent effect that optical flow methods have. However it is also noticed that at faster speeds, CNN methods have a tendency to make the objects disappear, leading to a video that doesn't make sense. This effect can be negated if the videos are on higher fps as there would be more context to the movement of the objects, thus making CNN methods more preferable on higher fps.

The effects of the change of speed are unpredictable. On MSE and LPIPS, the effects are irregular, whereas the PSNR generally shows a better result at slower speed.

However, the change from higher speed to a slower speed generally favors optical flow methods more as optical flow methods show a higher degree of change as seen above.

To a small extent, CNN methods perform better than optical flow methods on lower fps. While optical flow has the upper edge on objectivity metrics, CNN methods perform better in terms of visual clarity. Theoretically, CNN methods may have a better performance on higher fps compared to optical flow methods due to it having more contexts to the frames.

6.1. Future research opportunities

While this paper shows the effect of speed, and resolution on the objectivity metrics, it would be interesting to test the effects of higher fps on the performance of the methods. The idea of the effects on the metrics can be made clearer furthermore by using the same videos but manipulated to give different speeds as it can then give a more normalized view on the resulting video. Notably, this research is only limited to four different interpolation programs, two of which are created by the same person, Nicklaus et al, so increasing the amount of programs can increase the accuracy of the result.

7. Bibliography

- [1] X.-L. Yin *et al.*, “Reducing the X-ray radiation exposure frequency in cardio-angiography via deep-learning based video interpolation,” *arXiv.org*, Jun. 01, 2020.
<https://arxiv.org/abs/2006.00781> (accessed Jun. 11, 2023).
- [2] F. Reda, J. Kontkanen, E. Tabellion, D. Sun, C. Pantofaru, and B. Curless, “FILM: Frame Interpolation for Large Motion,” *arXiv:2202.04901 [cs]*, Jul. 2022, Accessed: Jun. 12, 2023. [Online]. Available: <https://arxiv.org/abs/2202.04901>
- [3] J. Choi and I. S. Kweon, “Deep Iterative Frame Interpolation for Full-frame Video Stabilization,” *arXiv.org*, Sep. 05, 2019. <https://arxiv.org/abs/1909.02641> (accessed Jun. 11, 2023).
- [4] A. S. Parihar, D. Varshney, K. Pandya, and A. Aggarwal, “A comprehensive survey on video frame interpolation techniques,” *The Visual Computer*, Jan. 2021, doi: <https://doi.org/10.1007/s00371-020-02016-y>. (accessed Oct. 25, 2023)
- [5] D. Gu, Z. Wen, W. Cui, R. Wang, F. Jiang, and S. Liu, “Continuous Bidirectional Optical Flow for Video Frame Sequence Interpolation,” Jul. 2019, doi: <https://doi.org/10.1109/icme.2019.00304>. (accessed Sep. 1, 2023)
- [6] Young Jin Oh and I.-K. Lee, “Two-step Temporal Interpolation Network Using Forward Advection for Efficient Smoke Simulation,” *Computer Graphics Forum*, vol. 40, no. 2, pp. 355–365, May 2021, doi: <https://doi.org/10.1111/cgf.142638>. (accessed Jun. 11, 2023).
- [7] R. Owens, “Optical flow,” *homepages.inf.ed.ac.uk*.
https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html (accessed Oct. 20, 2023)
- [8] A. Rajwade, “Optical Flow CS 763,” 2017. Accessed: Oct. 20, 2023. [Online]. Available: https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/OpticalFlow.pdf
- [9] W. Nur and Kamarul Hawari Ghazali, “Motion Detection Using Horn-Schunck Optical Flow,” *Communications in computer and information science*, pp. 202–209, Jan. 2012, doi: https://doi.org/10.1007/978-3-642-35594-3_29. (accessed May. 5, 2023)

- [10] First Principles of Computer Vision, “Lucas-Kanade Method | Optical Flow,” *www.youtube.com*, May 03, 2021. <https://www.youtube.com/watch?v=6wMoHgpVUn8> (accessed Sep. 03, 2023).
- [11] K. Kitani, “Lucas-Kanade Optical Flow,” CMU School of Computer Science, <https://www.cs.cmu.edu/~16385/s15/lectures/Lecture21.pdf> (accessed Jul. 22, 2023).
- [12] R. Rojas, “Lucas-Kanade in a Nutshell.” Accessed: Sep. 08, 2023. [Online]. Available: https://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/tutorials/Lucas-Kanade2.pdf
- [13] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, Aug. 1981, doi: [https://doi.org/10.1016/0004-3702\(81\)90024-2](https://doi.org/10.1016/0004-3702(81)90024-2). (accessed Sep. 6, 2023)
- [14] LearnOpenCV, “Understanding Convolutional Neural Networks: A Complete Guide,” *learnopencv.com*, Jan. 18, 2023. <https://learnopencv.com/understanding-convolutional-neural-networks-cnn/> (accessed Dec. 23, 2023).
- [15] IBM, “What is Deep Learning?,” *www.ibm.com*, 2023. <https://www.ibm.com/topics/deep-learning> (accessed Jul. 18, 2023).
- [16] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, “Depth-Aware Video Frame Interpolation,” *arXiv:1904.00830 [cs]*, Apr. 2019, Accessed: Jul. 18, 2023. [Online]. Available: <https://arxiv.org/abs/1904.00830>
- [17] M. Mishra, “Convolutional Neural Networks, Explained,” *Medium*, Aug. 27, 2020. <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939> (accessed Jul. 17, 2023).
- [18] K. Ghodoussi, N. Sheth, Z. Durante, and M. Wagner, “Deep CNN Frame Interpolation with Lessons Learned from Natural Language Processing,” *arXiv.org*, Sep. 16, 2018. <https://arxiv.org/abs/1809.05286> (accessed Jul. 18, 2023).
- [19] “Introduction to Motion,” *BYJUS*, May 15, 2019. <https://byjus.com/physics/introduction-to-motion/> (accessed Sep. 27, 2023).

- [20] D. Brunner, “Frame Rate: A Beginner’s Guide | Blog | TechSmith,” *Welcome to the TechSmith Blog*, Jul. 11, 2019.
<https://www.techsmith.com/blog/frame-rate-beginners-guide/> (accessed Sep. 27, 2023).
- [21] U. Sara, M. Akter, and M. S. Uddin, “Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study,” *Journal of Computer and Communications*, vol. 07, no. 03, pp. 8–18, 2019, doi: <https://doi.org/10.4236/jcc.2019.73002>. (accessed Sep. 9, 2023)
- [22] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric,” *arXiv:1801.03924 [cs]*, Apr. 2018, Accessed: Dec. 07, 2023. [Online]. Available: <https://arxiv.org/abs/1801.03924>.
- [23] S. T. Welstead, *Fractal and wavelet image compression techniques*. Bellingham, Wash.: Spie Optical Engineering Press, 1999. (accessed Dec. 11, 2023)

7.1. Programs used

CNN Programs

Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, “Real-time intermediate flow estimation for video frame interpolation,” *arXiv.org*, <https://arxiv.org/abs/2011.06294> (accessed Oct. 4, 2023).

S. Niklaus, L. Mai, and F. Liu, “Video frame interpolation via adaptive separable convolution,” *arXiv.org*, <https://doi.org/10.48550/arxiv.1708.01692> (accessed Sep. 3, 2024).

Optical programs

Dthpham, “sminterpolate,” GitHub, <https://github.com/dthpham/sminterpolate> (accessed Dec. 19, 2023).

S. Niklaus and F. Liu, “Softmax splatting for video frame interpolation,” arXiv.org, <https://doi.org/10.48550/arXiv.2003.05534> (accessed Dec. 17, 2023).

Code for getting dependent variable

prernaroy, “How to get the SSIM comparison score between two images?,” *Stack Overflow*, Mar. 22, 2022.

<https://stackoverflow.com/questions/71567315/how-to-get-the-ssim-comparison-score-between-two-images> (accessed Dec. 20, 2023).

R. Zhang, “richzhang/PerceptualSimilarity,” *GitHub*, Sep. 16, 2023.

<https://github.com/richzhang/PerceptualSimilarity> (accessed Dec. 20, 2023).

To be able to get the values of the metrics between the interpolated frames and groundtruth, several external libraries will be used, namely, Numpy, Scikit, and LPIPS.

```
from math import log10, sqrt
from os import listdir
import lpips
import cv2
import numpy as np
import openpyxl
from skimage.metrics import structural_similarity
```

```
wb = openpyxl.load_workbook('Datas.xlsx')
ws = wb['Medium240p'] # Speed, resolution
dir1= input("path/to/files/*.png")
dir2 = input("path/to/files/*.png")
```

```
loss_fn = lpips.LPIPS(net='alex')
```

```
class PSNRMSEValue:
    def __init__(self, original, compressed):
        self.mse = np.mean((original - compressed) ** 2)
        if self.mse == 0:
            self.psnr = 100
        max_pixel = 255.0
        self.psnr = 20 * log10(max_pixel) - 10 * log10(self.mse)

for groundt, frames, i in zip(listdir(dir1), listdir(dir2), range(5, 48)):
    print(frames)
    before = cv2.imread(dir1 + "/" + groundt)
    after = cv2.imread(dir2 + "/" + frames)

    img0 = lpips.im2tensor(lpips.load_image(dir1 + "/" + groundt))
    img1 = lpips.im2tensor(lpips.load_image(dir2 + "/" + frames))

    # Convert images to grayscale
    before_gray = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
    after_gray = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

    # Compute SSIM between two images
    (score, diff) = structural_similarity(before_gray, after_gray, full=True)

    values = PSNRMSEValue(before_gray, after_gray)

    print("Image similarity ||", score)
    print("PSNR value    ||", values.psnr, " dB")
    print("MSE value     ||", values.mse, " dB")
    dist01 = loss_fn.forward(img0, img1)

    print("LPIPS value    || %.3f"%dist01)
    print("\n")

# Will change depending on the type of motion and software
ws[f'BA{i}'] = score
ws[f'BB{i}'] = values.psnr
ws[f'BC{i}'] = values.mse
ws[f'BD{i}'] = float(" %.3f"%dist01)

wb.save('Datas.xlsx')
```

Code for getting the difference mask

```
from skimage.metrics import structural_similarity
import cv2

before = cv2.imread('DIRECTORY')
after = cv2.imread('DIRECTORY')

# Convert images to grayscale
before_gray = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
after_gray = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

# Compute SSIM between two images, get the difference mask for the two image
(score, diff) = structural_similarity(before_gray, after_gray, full=True)
print(f"Image similarity: {score}")

diff = (diff * 255).astype("uint8")
cv2.imwrite('FILENAME', diff)
cv2.imshow('before', before)
cv2.imshow('after', after)
cv2.imshow('diff', diff)
cv2.waitKey(0)
```